Software Engineering Challenges for Parallel Processing Systems

Lt Col Marcus W Hervey, USAF
AFIT/CIP
marcus.hervey@us.af.mil

Disclaimer

"The views expressed in this presentation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government."

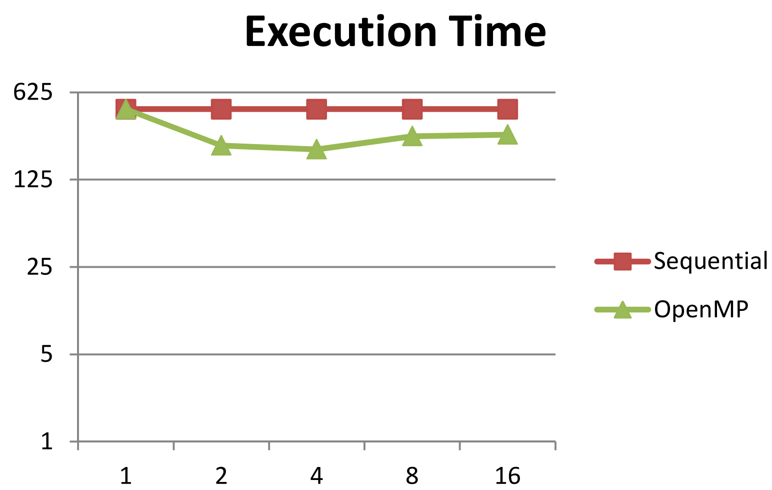
Outline

- Motivation
- A Brief Overview of Parallel Computing
- Parallel Programming Challenges
- The Need for Parallel Software Engineering
- Research Directions
- Summary

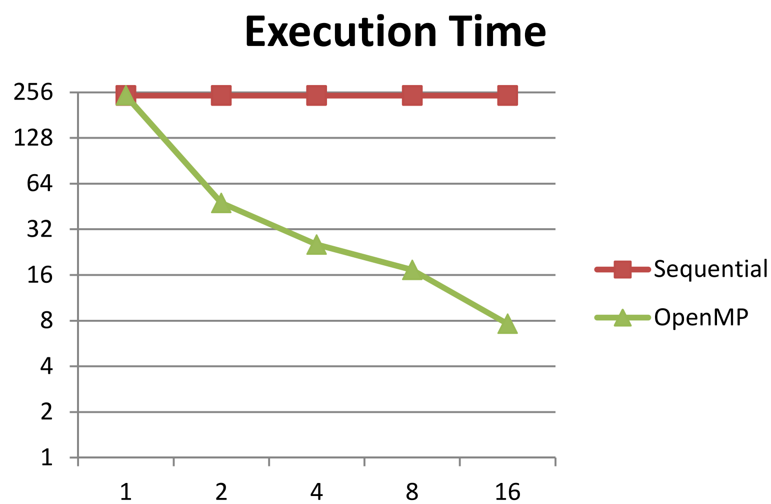
From Moore’s to Cores

- Before sequential programs were made faster by running on higher frequency computers without changes to the code
- Chip manufacturers ran into problem with continuing down this path
  - Heat generation
  - Power consumption
- Redefined metric from processor speed to performance (# of processors/cores)
- Today optimum performance will require significant code changes with the knowledge to develop correct and efficient parallel programs

What's All the Fuss About?



Matrix Multiply using OpenMP



Jacobi using OpenMP

Parallel Processing:

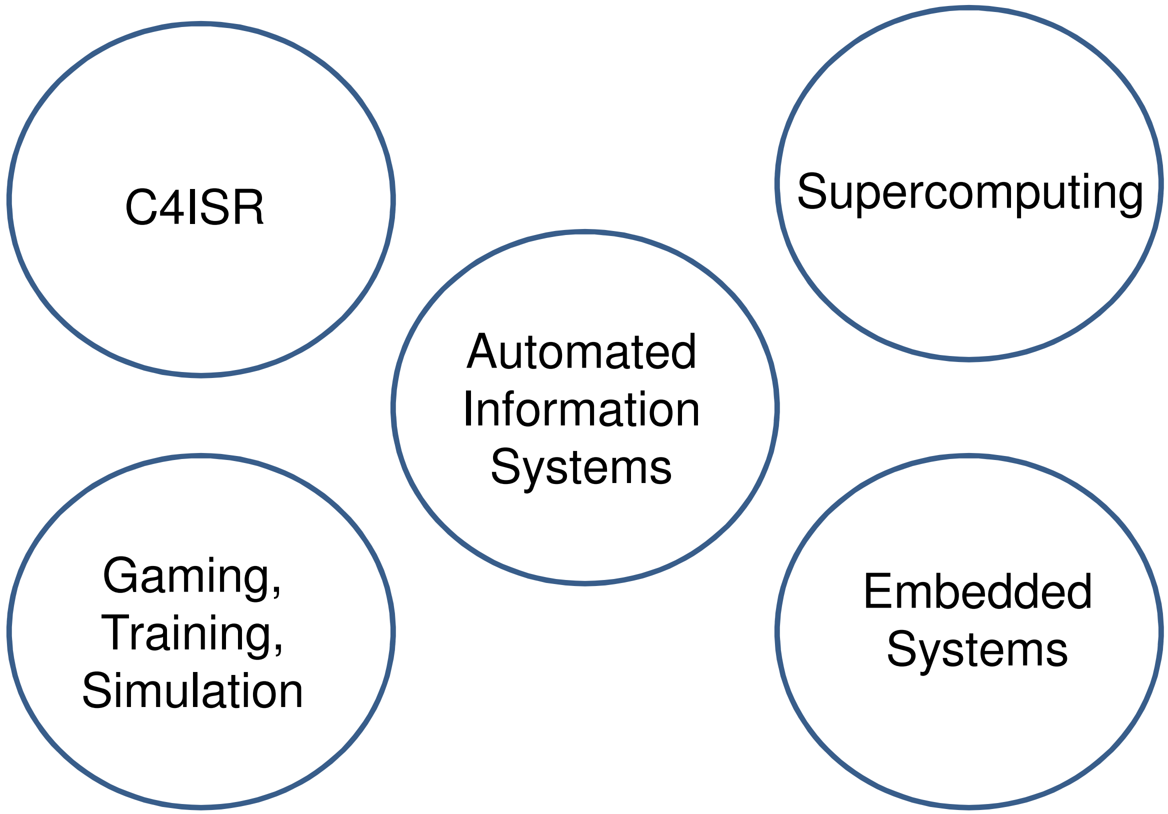
- Solves problems faster or solves larger problems
- More complex -- Must match best algorithm with best programming model and best architecture

Applications of Parallel Computing

- Embedded Systems
  - Cell phones, Automobiles, PDAs
- Gaming Systems
  - Playstation 3, Xbox 360
- Desktop/Laptops
  - Dual-core/Quad-core
- Supercomputing (HPC/HPTC/HEC)
  - www.top500.org

Parallel Processing is mainstream!

Military Applications of Parallel Computing

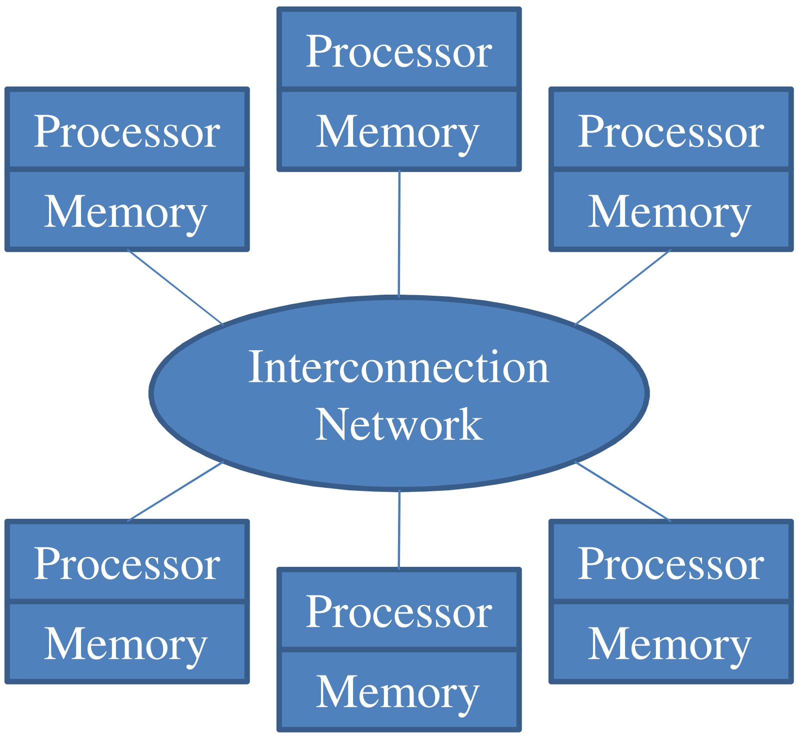


The New Frontier

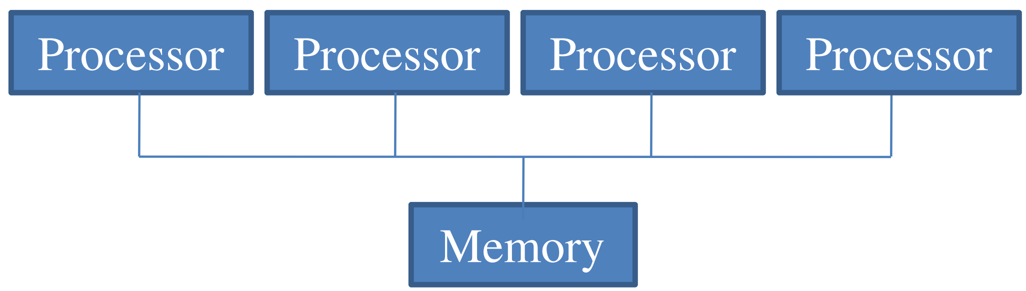
- **Standard Architectures**
  - Beowulf Clusters / Grid Computing
  - Dual-core/Quad-core – Intel/AMD
  - Intel’s 80-core machine
- **Non-standard Architectures**
  - 72-core machine – Sicortex
  - FPGAs - Field-programmable gate array
  - GPGPUs – Nvidia, AMD (ATI)
  - Cell Processor – IBM – Playstation 3
  - Accelerators - Clearspeed

Parallel Processing Architectures

Distributed Memory



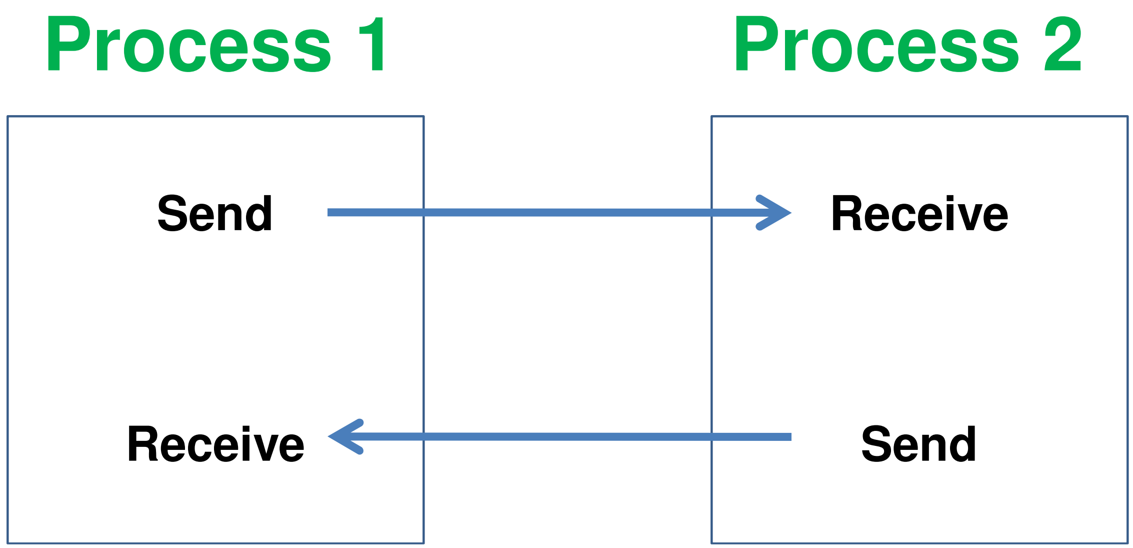
Shared Memory



…there is also Distributed Shared Memory

Message Passing Model

Communicates by sending/receiving messages



- **OpenMPI**
- **MPICH**

Designed for Distributed Memory Machines

OpenMPI Code Example

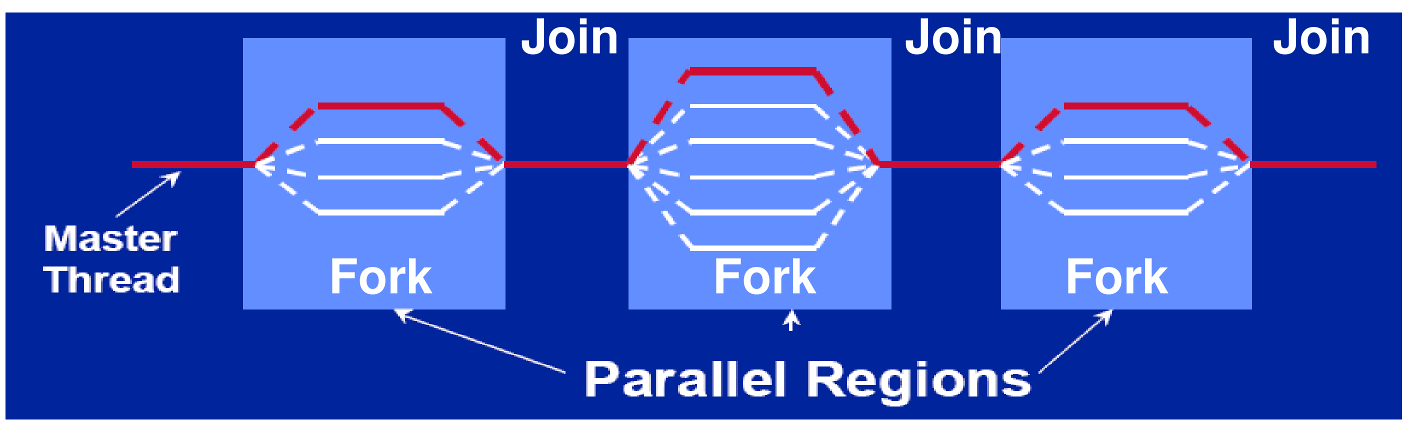
```
#include <stdio.h>
 #include <mpi.h>
int main(int argc, char **argv) {
   char buff[20];  int myrank;
   MPI_Status status;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
   if (myrank == 0) {
   strcpy(buff, “Hello World!\n”);
   MPI_Send(buff,20,MPI_CHAR,1,99,MPI_COMM_WORLD);
   }
   else {
   MPI_Recv(buff,20,MPI_CHAR,0,99,MPI_COMM_WORLD,&status);
      printf(“received :%s:\n”, buff);
   }
   MPI_Finalize();
return 0;
}
```

Shared-Memory Model

Communicates by accessing shared memory

- **OpenMP programming model**
- **POSIX Theads (Pthreads)**
- **Unified Parallel C**

OpenMP Fork-join Pattern



OpenMP Code Example

Without OpenMP

```
#include<stdio.h>

int main(void)
{
printf("Hello World!\n");
return 0;
}
```

With OpenMP

```
#include <stdio.h>
 #include <omp.h>
int main(void)
{
 int threadid = 0;
 #pragma omp parallel private(threadid)
 {
 threadid = omp_get_thread_num();
 printf("%d : Hello World!\n", threadid);
 }
 return 0;
}
```

- Implemented as C/C++/Fortran language extensions
- Composed of compiler directives, user level runtime routines, environment variables
- Facilitates incremental parallelism

Pthreads Code Example

```
#include <stdio.h>
 #include <pthread.h>
define NUM_THREADS 5

void *HelloWorld(void *threadid) {
   printf(“%d : Hello World!\n”, threadid);
   pthread_exit(NULL);
}

int main() {
 pthread_t threads[NUM_THREADS];
 int rc, t;
 for (i=0; i<NUM_THREADS; i++) {
    printf(“%d : Hello World!\n”, i);
    rc = pthread_create(&threads[i], NULL, HelloWorld, (void *) t);
    if (rc) {
      printf(“ERROR; return code from pthread_create() is %d\n”, rc);
      exit (-1);
    }
 }
 pthread_exit(NULL);
}
```

UPC Code Example

Without UPC

```
#include<stdio.h>

int main(void)
{
printf("Hello World!\n");
return 0;
}
```

With UPC

```
#include <stdio.h>
 #include <upc.h>
int main(int argc, char *argv[])
{
  int i;

  for(i=0; i<THREADS; i++)
  {
     if (i==MYTHREAD)
     {
        printf("%d : Hello World!\n", MYTHREAD);
      }
return 0;
}
```

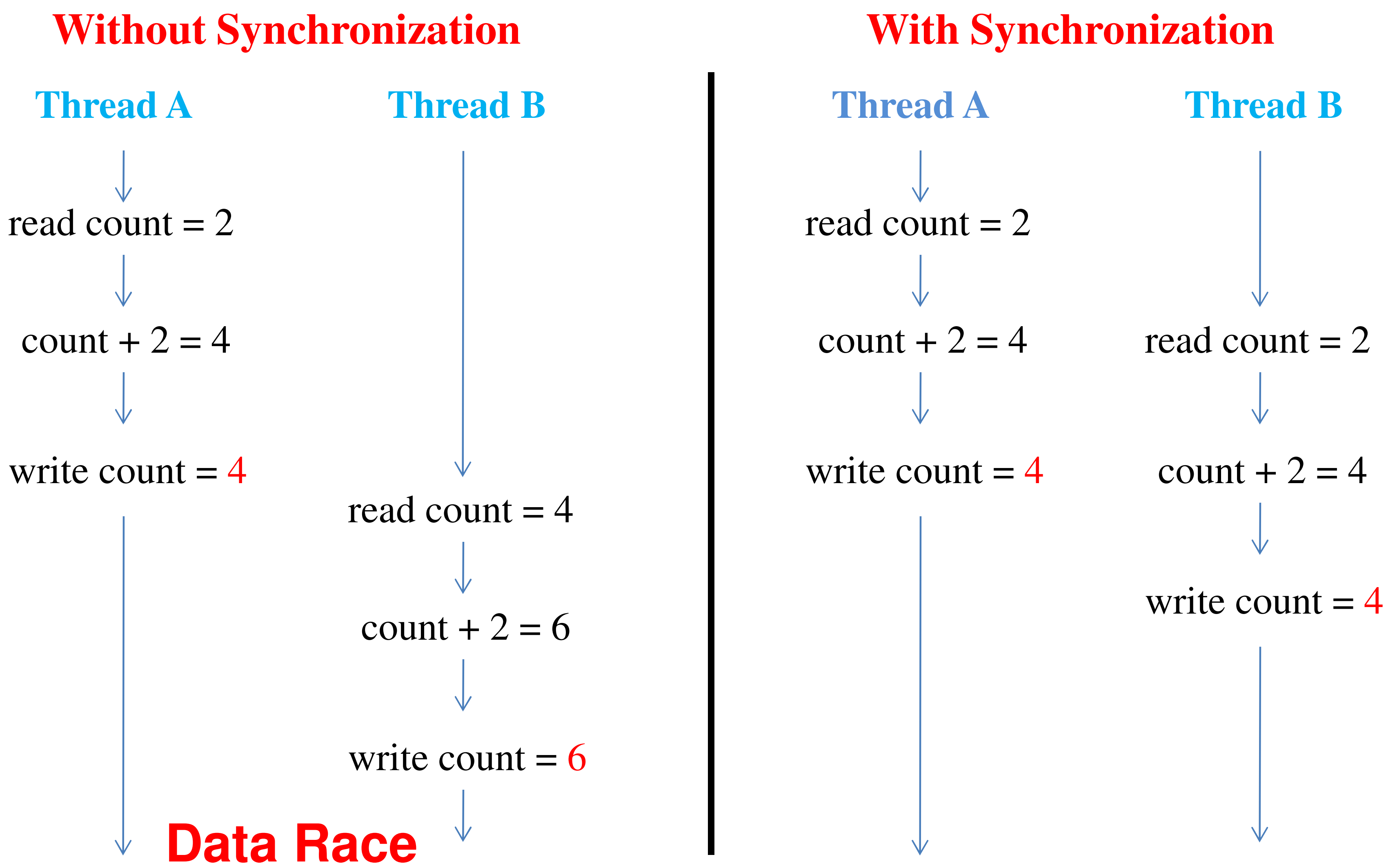
Major Parallel Programming Challenges

- **Parallel Thinking/Design**
  - Identifying the parallelism
  - Parallel algorithm development
- **Correctness**
  - Characterizing parallel programming bugs
  - Finding and removing parallel software defects
- **Optimizing Performance**
  - Maximizing speedup and efficiency
- **Managing software team dynamics**
  - Complex problems require large, dispersed, multi-disciplinary teams

A Different Species of Bugs

- **Data Races**
  - When an interleaving of threads results in an undesired computation result
- **Deadlock**
  - When two or more threads stop and wait for each other
- **Priority Inversion**
  - A higher priority thread is preempted by a lower priority thread
- **Livelock**
  - When two or more threads continue to execute, but make no progress toward the ultimate goal
- **Starvation**
  - When some thread gets deferred forever

Data Race Example



This type of error caused by Therac-25 radiation therapy machine resulted in 5 deaths

Deadlock

MPI Example

PROCESS 1

Send (Processor 2)
Receive(Processor 2)

Waiting on Process 2 to receive message

PROCESS 2

Send(Processor 1)
Receive(Processor 1)

Waiting on Process 1 to receive message

OpenMP Example

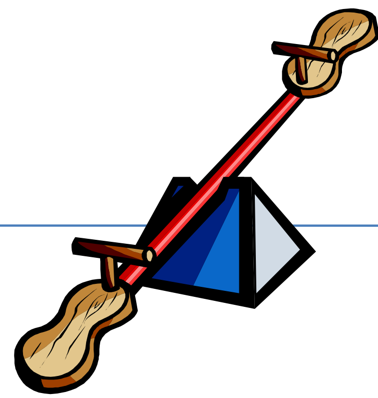
```
worker () {
#pragma omp barrier
}

main () {
#pragma omp parallel sections
{
#pragma omp section
  worker();
  }
}
```

Synchronization Errors

Not Enough

Too Much



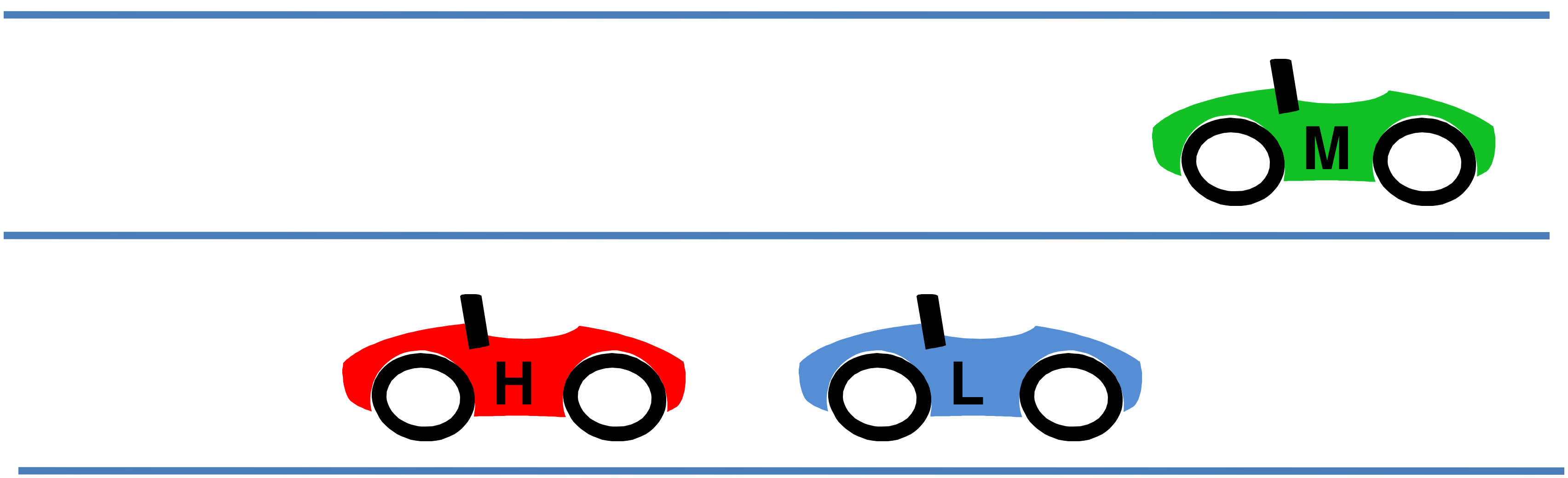
Data Races

Deadlock

- Missing or inappropriately applying synchronization can cause data races
- Applying too much synchronization can cause deadlock

Priority Inversion

- Lower priority thread preempts higher priority thread
  - Low-priority thread enters critical section.
  - High-priority thread wants to enter critical section, but can't enter until low-priority thread is complete.
  - Medium-thread pre-empts higher priority thread
- This type of error caused Mars Pathfinder failure

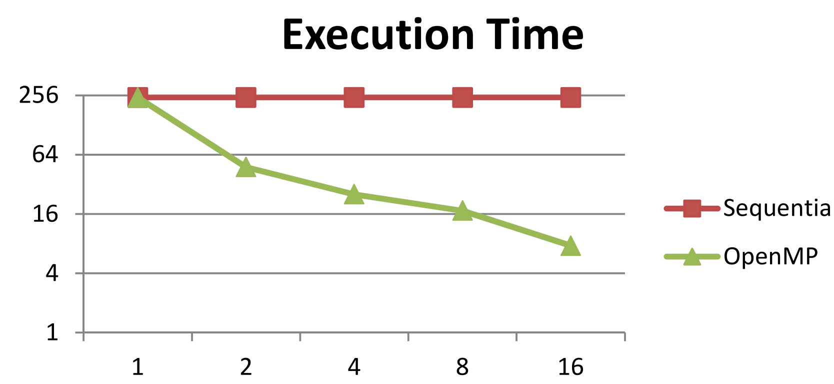


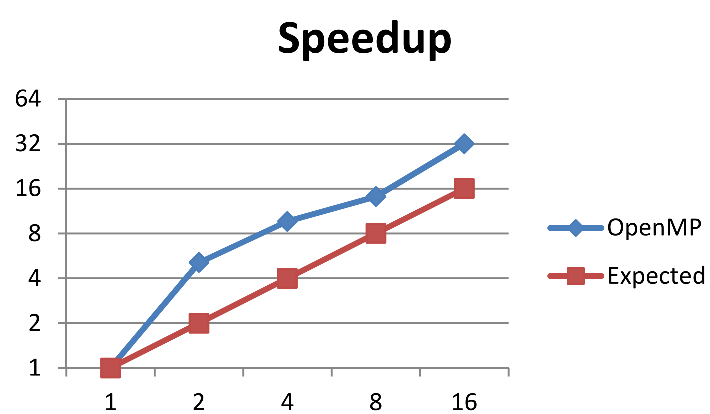
Parallel Performance

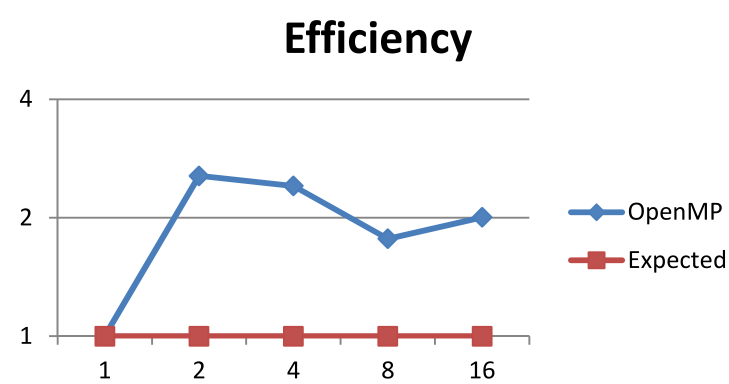
- **Execution time**
  - Time when the last processor finishes its work
  - Amdahl's Law – Sequential portions of code limit speedup
    - Most parallel codes have sequential portion(s)
- **Speedup**
  - (1 CPU execution time)/ (P CPUs execution time)
  - Must compare to the best sequential algorithm
- **Efficiency**
  - Speedup/P
  - 100% efficiency is hardly ever possible

Parallel Performance Metrics

Performance of Jacobi using OpenMP







For optimum performance, parallel developers need to have an understanding of the application and the architecture

Parallel Software Quality Goals

- Correctness, Robustness and Reliability
- Performance
  - Speedup, Efficiency, Scalability, Load Balance
- Predictability – Cost, Schedule, Performance
  - Managing complexity of harder problems with more non-standard architectures and more diverse teams
- Maintainable

Lack of Parallel Software

HPC wire

The Leading Source for Global News and Information Covering the Ecosystem of High Productivity Computing / April 7,

Home Page | Free Subscription | Advertising | About HPCwire

Software:

Council on Competitiveness Study Reveals Lack of Software

The Council on Competitiveness, a national organization of business, academic and labor executives, has released the second part of a study that reveals that the lack of scalable application software is preventing many companies from using high performance computing (HPC) more aggressively for competitive advantage. Part B of the Council on Competitiveness Study of ISVs Serving the High Performance Computing Market concludes that major U.S. industries often cannot get the application software they need to drive innovation and global competitiveness. Both parts of the pioneering study were sponsored by the Defense Advanced Research Projects Agency (DARPA) and conducted by leading market research firm IDC.

Part A of this study revealed that the independent software vendor (ISV) business model for developing advanced application software for HPC has nearly evaporated, and that ISVs must focus most of their software development of the broader commercial market.

"This study demonstrates that the lack of production quality HPC application software is a soft spot in the competitiveness armor of the U.S." said Council on Competitiveness President Deborah L. Wince-Smith. "When U.S. industries can not obtain the application software they want and need, innovation is stymied and competitiveness is compromised. Fortunately, we are finding that most ISVs and a substantial portion of U.S. businesses are willing to partner with each other, as well as universities and national laboratories to speed progress in addressing this challenge."

"Part B: End User Perspectives" directly surveyed a select group of highly experienced HPC users in U.S. businesses, representing a wide range of industries, from defense to entertainment to consumer products. The study revealed the U.S. business requirements for advanced HPC application software, and the financial and technical obstacles blocking firms from obtaining it. The perspectives given by these experienced users echoed many of the findings from the Council's recently released software workshop report "Accelerating Innovation for Competitive Advantage: The Need for HPC Application Software Solutions."

A comparison of the key findings from Parts A and B is found in the following chart. The findings reveal the need for more aggressive use of HPC in American business and the current plans ISVs have to meet these needs. The limitations of HPC-specific ISV application software are not the only barrier to fuller exploitation of HPC but are regularly cited by industrial end users as the most important constraint.

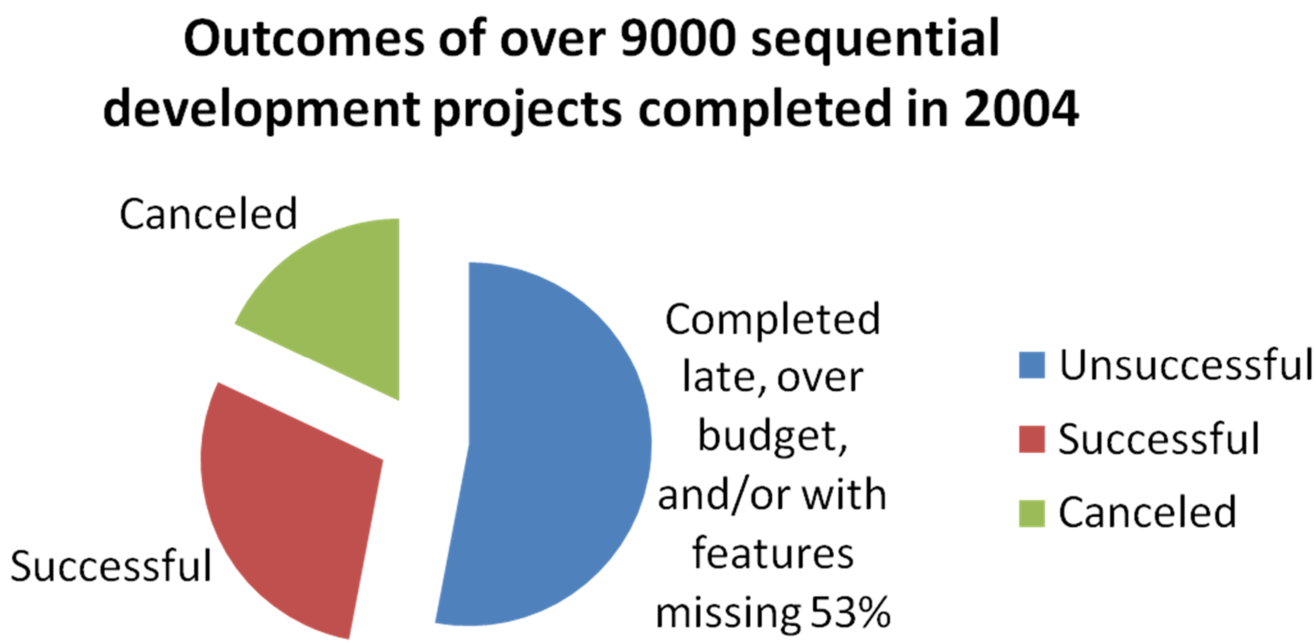
Study Part A: Current ISV Market Dynamics

- The business model for HPC- specific application software has all but evaporated in the last decade.
- ISV applications can exploit only a fraction of the problem-solving power of today's high-performance computers.
- For many applications, the ISVs know how to improve scalability but have no plans to do so because the HPC market is too small to justify the R&D investment.
- There is a lack of readiness among ISV suppliers for petascale systems.
- Market forces alone will not address the gap between HPC users' needs and ISV software capabilities.
- Most ISVs would be willing to partner with outside parties to accelerate application software development.

Study Part B: HPC End Users' Perspectives

- HPC-specific ISV application software is indispensable for U.S. industrial competitiveness.
- Virtually all of the firms said they have larger problems that they can't solve today.
- The lack of scalable application software is preventing many industrial users from using HPC more aggressively for competitive advantage.

The Need for Software Engineering



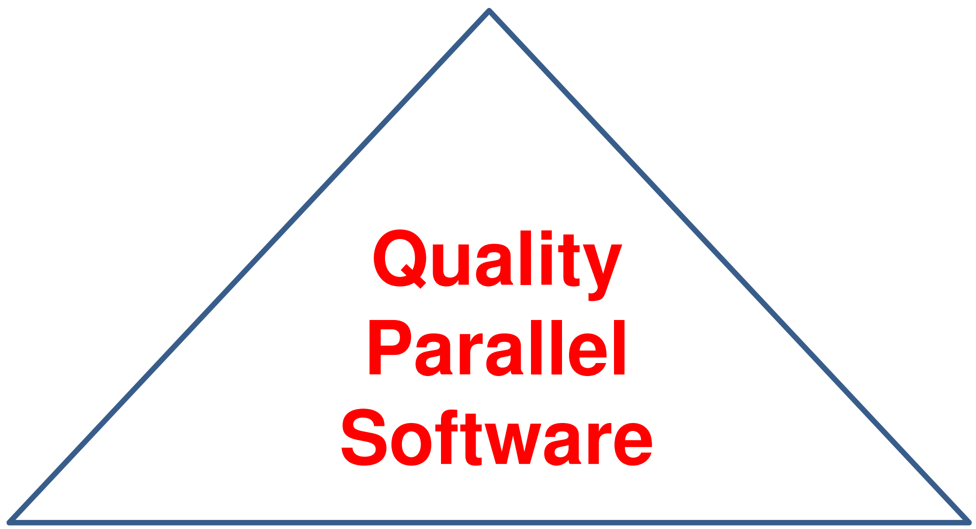
Source: [Hayes, Frank, "Chaos is Back," Computerworld, November 8, 2004.]

Software engineering is needed to create an environment for the development of quality parallel software (reliable, predictable and maintainable)

Parallel Software Engineering

Process

Defined, Repeatable



Technology

Eclipse Parallel Tools Platform,
Thread Analyzer, Thread Checker,
DDT, Totalview

People

Technical and ProcessTraining,
Discipline

Result : Predictable Cost, Schedule and Performance

Software Life Cycles

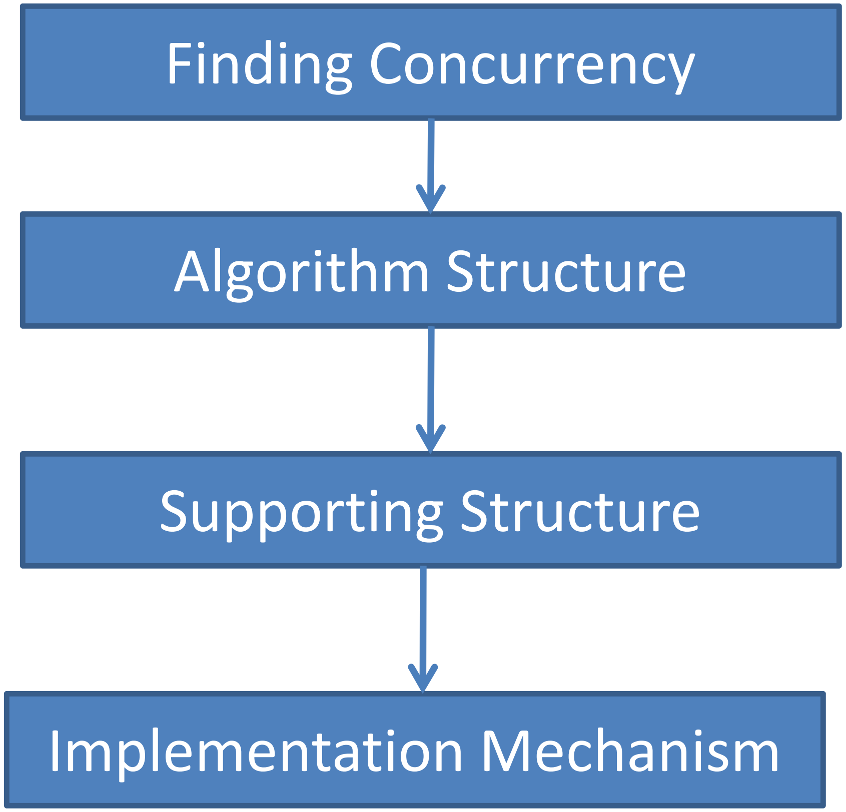
Sequential
Development Methodology

- Requirements Analysis
- Design
- Implementation
- Testing
- Deployment

Parallel
Development Methodology

- Requirements Analysis
- Design
- Sequential Implementation
- Code Profiling
- Parallel Design
- Parallel Implementation
- Testing
- Code Optimization (Tuning)
- Deployment

Patterns for Parallel Programs



- Decomposing the problem to exploit concurrency
- Structuring the algorithm by tasks, data decomposition or by flow of data
- Defining the shared data structures that support algorithm implementation
- Implementing management, communication and synchronization

Source: [T. A. Mattson, B. Sanders and B. Massingill. Patterns for Parallel Programming, 2004.]

Technology

- **Parallel Languages**
  - OpenMPI, OpenMP, UPC, POSIX, X10, Fortress, Chapel
- **Compilers**
  - Intel, Sun, Open64
- **IDEs**
  - Eclipse Parallel Tools Platform
- **Debugging Tools**
  - TotalView, DDT, Thread Checker, Thread Analyzer
- **Performance Tools**
  - PAPI, TAU

People

- Understand standard/non-standard architectures
- Learn parallel programming/bug patterns
- Comprehend parallel language strengths/weaknesses
- Learn the process and tools
- Work within multi-disciplinary teams

Research Directions

- **Exploiting Nonstandard Architectures**
  - Cell Processors, GPGPUs, FPGAs, accelerators
- **Parallel Programming Models**
  - Extending existing languages C, C++, and Fortran
  - New languages development: X10, Chapel, Fortress
  - Hybrid code development (OpenMP/MPI)
- **Parallel Compilers**
  - Code optimization and auto-parallelization
- **Productivity Enhancing Tools**
  - IDEs, profiling, optimization and debugging tools

Resources

- B. Chapman, G. Jost, and R. Van Der Pas. **Using OpenMP: Portable Shared Memory Parallel Programming**. The MIT Press, 2008.
- T. G. Mattson, B. A. Sanders, and B. L. Massingill. **Patterns for Parallel Programming**. Addison-Wesley Professional, 2004.
- cOMPunity, www.compunity.org
- DoD HPCMO, www.hpcmo.hpc.mil
- HPC Bug Base, www.hpcbugbase.org
- HPC Tools Group, http://www2.cs.uh.edu/~hpctools/
- OpenMP, www.openmp.org
- OpenMPI, www.open-mpi.org

Summary

- Parallel computing is all around you!
- Parallel programming introduces more complex software defects that are hard to detect and debug
- Parallel software performance requires attention to issues of communications, synchronization, scalability and load balance
- Better processes, tools and training are needed to improve the practice and predictability of parallel software engineering
- Software developers and acquisition personnel should be aware of the opportunities and challenges of parallel software

For More Information

Lt Col Marcus W Hervey, USAF

AFIT/CIP

marcus.hervey@us.af.mil

www.marcushervey.com

Acronym List

- C4ISR – Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
- DDT – Distributed Debugging Tool
- FPGA – Field Programmable Gate Array
- GPGPU – General Purpose Graphics Processing Unit
- HPC – High-Performance Computing
- IDE – Integrated Development Environment
- MPICH – Message Passing Interface Chameleon
- OpenMP – Open Mulit-Processing
- OpenMPI – Open Message Passing Interface
- PAPI – Performance Application Programming Interface
- TAU – Tuning and Analysis Utilities
- UPC – Unified Parallel C